REMARKS

This amendment responds to the Office Action mailed on November 15, 2004. Filed concurrently herewith is a *Request for a Three Month Extension of Time* which extends the shortened statutory period for response to expire on May 15, 2005. Accordingly, Applicants respectfully submit that this response is being timely filed.

Claims 1-26 were pending. New claims 27-30 are submitted for examination on their merits. Accordingly, claims 1-30 are now pending in the present application, and Applicant believes these claims are in proper condition for allowance for the reasons set forth below.

Provisional Double-Patenting Rejections

Paragraphs 10 and 11 of the Office Action provisionally rejected claims 1-26 under 35 U.S.C. § 101 as claiming the same subject matter of that of claims 17-54 of co-pending U.S. Patent Applications Serial Nos. 10/164,789 and 10/165,457. Applicant respectfully submits that claims 1-26 of the present application are in condition for allowance based on the remarks below. Thus, Applicant respectfully submits that these double patenting rejection are premature and moot since both of the '789 and '457 applications are pending and have not issued as a U.S. patent. Accordingly, Applicants respectfully request withdrawal of these provisional rejections and allowance of the pending claims in the present application.

Claim Rejections Under 35 U.S.C. §102

Paragraph 13 of the Office Action rejects claims 13, 19 and 24 under U.S.C. § 102(e) as being anticipated by U.S. Patent No. 5,768,593 to Walters et al. (hereinafter "Walters"). Applicant respectfully traverses this rejection and submits that the claims at issue are patentable over those patents for the following reasons.

INDEPENDENT CLAIM 13

According to a pertinent aspect of Applicant's disclosure to which Claim 13 is directed, translation of a given portion of program code instructions may, for example, yield two different sets of target code instructions, depending upon the particular set of entry conditions which exist at the time that the given portion of program code is encountered. One may refer to the two sets

of entry conditions as the "prevailing set" and the "subsequent set" of conditions. The "subsequent set" of conditions will be synonymously referred to hereafter as the "second set" of conditions.

According to one aspect of Applicant's method of generating target code, upon first encountering the given portion of program code, only the target code necessary to execute that portion of program code under the "prevailing set" of entry conditions is initially generated and stored. When the same portion of program code is subsequently encountered, a determination is made as to whether target code corresponding to the "second set" of conditions ("the subsequent conditions") has been generated. If not, additional target code necessary to execute the same portion of program code under that second set of conditions is generated.

Subject code instructions may have a multiple number of possible effects or functions, not all of which may be required when the subject code instruction is first executed, and some of which may not in fact be required at all. Prior art translators, like Walters, required the generation of complex target code instructions that were capable of representing all possible entry conditions and further capable of performing all of the effects and functions of the subject code instructions, as described on pages 11-13 of the present specification.

The method embodied in Claim 13 of the present application overcomes these problems by translating only those features of the subject code instructions required for the particular conditions at that time. This provides a significant performance advantage, since the amount of code to be translated for execution is reduced. Upon initial translation of a given portion of program code, only the target code is generated which is required to execution that portion of the program code with a "prevailing set of conditions." For example, only target code associated with the most-frequently executed effects or functions of the subject code instructions may be generated. If, when the same portion of subject code is subsequently encountered, additional functionality is required (e.g., for a less-frequently used functionality) for the subsequent conditions at that time, then additional target code is generated to executed the portion of the subject code requiring this additional or different functionality.

To support an "anticipation" rejection under 35 U.S.C. § 102, the reference must teach every element and recitation of the Applicant's claims. Rejections under 35 U.S.C. § 102 are

proper only when the claimed subject matter is identically disclosed or described in the prior art. Thus, the Walters reference must clearly and unequivocally disclose every element and recitation of the claimed invention in claims 13, 19 and 24 in order to support a rejection under U.S.C. § 102(e).

A. Walters Fails to Teach or Suggest Multiple Sets of Target Code for the Same Portion of Subject Code

As set forth above, Claim 13 recites that two sets of target code are generated for the same portion of subject code: 1) generating only target code which is required to execution that portion of the program code with a prevailing set of conditions (i.e., the first set of target code) on the initial translation of a given portion of the program code, and 2) when subsequently entering the same portion of program code, generating additional target code (i.e., the second set of target code) required to execute the same portion of program code with subsequent conditions, if no target code has previously been generated and stored for that portion of program code for the subsequent conditions.

Referring to the Walters patent, and particularly to column 7 cited by the Office Action, there is no disclosure of the concept of generating different sets of target code for the same portion of program code (e.g., the same sequence of program code instructions). Walters merely teaches that a block of native code (i.e., target code) should be generated and stored in cache upon initial translation of the subject code. When encountering a non-native instruction (i.e., subject code instruction), Walters teaches in column 7, lines 17-23 that a look up is performed to see if a corresponding native code block is already stored in cache for the non-native instruction and, if so, the native block code in the code cache is executed. As can be seen, Walters simply teaches translating subject code into target code when the subject code instruction is initially encountered, where the same corresponding target code for this subject code instruction is always retrieved from cache for execution whenever this subject code instruction is encountered again. Walters fails to teach or suggest generating additional target code corresponding to the subsequent conditions when subsequently entering the same portion of subject code for which target code has already been generated upon initial translation.

Since claim 13 clearly recites generating different target code for, inter alia, the same "given portion of program code," claim 13 is clearly not anticipated by Walters et al. Accordingly, Applicants respectfully submit that the Section 102 rejection of claim 13 and its respective dependent claims should be withdrawn and the claims allowed.

B. Examiner's Response to Applicant's Arguments

In paragraph 3 of the Office Action, the Examiner contends that Applicant is relying upon features of his invention that are not recited in the claims when Applicant asserts that the claims recite generating different sets of target code for the same portion of program code. Applicant respectfully traverses the Examiner's contention, because, as set forth above, claim 13 clearly recites generating two sets of target code for the same portion of subject code: 1) the target code which is generated for the prevailing set of conditions upon initial translation of the program code, and 2) the additional target code that is generated for the subsequent conditions when the same subject code is subsequently entered. Thus, claim 13 clearly recites generating multiple sets of target code for the same portion of subject code, one set for the prevailing set of conditions upon initial translation and other sets for the different subsequent conditions when the subject code is subsequently encountered and the subsequent conditions are different from the prevailing conditions.

C. Walters Fails to Teach or Suggest Generating Additional Target Code When Program Code is Subsequently Entered

It is asserted in paragraph 13 of the Office Action that Walters discloses both 1) initially translating a given portion of program code into target code, and 2) when the same portion of program code is subsequently entered, subsequently generating target code for the program code if the target code has not been previously generated. However, Walters only discloses a single translation of a portion of program code, not multiple translations of the same portion of program code. The Examiner is incorrectly construing Walters' disclosure of performing this single translation of program code into target code as satisfying the multiple translations performed in the method recited in claim 13.

In Walters, when a non-native instruction is encountered, that instruction is treated as an 'entry point instruction' and a hash table lookup procedure is used to look up the address of the

entry point instruction to determine if a corresponding native code block has already been translated and stored in the code cache. If so, the native code block in the code cache is executed. *See column 7, lines 16-23.* Thus, after a non-native instruction is encountered and translated, it will be stored in the code cache and this same translated code will then be used whenever this non-native instruction is encountered again. It can clearly be seen from FIG. 3 of Walters that once a native code block is generated for a non-native instruction (FC instruction), a "YES" will always be returned in step 162 when that same non-native instruction is encountered again and the corresponding native code block in the code cache will always be executed in step 164. The flow of FIG. 3 only reaches step 170 where the native code (i.e., target code) is generated if a "NO" had been returned in step 162 meaning that the same non-native instruction has never been translated before.

Clearly, Walters only discloses making a single translation of a portion of program code after making a determination of whether target code for that same portion of program code had previously been generated. The "initial translation" of a given portion of program code into target code is performed in step 170 of FIG. 3 of Walters and there are no subsequent steps taught or suggested in Walters for generating additional target code in addition to the already initially translated target code when subsequently encountering the same portion of program code.

Since claim 13 clearly recites initially generating target code for a given portion of program code and subsequently generating additional target code for, inter alia, the same "given portion of program code," claim 13 is clearly not anticipated by Walters et al. Accordingly, Applicants respectfully submit that the Section 102 rejection of claim 13 and its respective dependent claims should be withdrawn and the claims allowed.

Claim Rejections Under 35 U.S.C. §103

Paragraph 15 of the Office Action rejects claims 1-5, 7-12, 15-18, 20-23, 25 and 26 under U.S.C. § 103(a) as being obvious over U.S. Patent No. 5,613,117 to Davidson (hereinafter "Davidson") in view of Walters. Applicant respectfully traverses this rejection and submits that the claims at issue are patentable over those patents for the following reasons.

CLAIMS 1 AND 15

Independent Claims 1 and 15 recite, inter alia, generating different intermediate representations for the same "given portion" of program code in response to respective "previous prevailing" and "subsequent" conditions. These conditions can be the dynamic conditions occurring during the translation, such as the current state of the abstract registers. As described on pages 26-27 of the present specification, in order for the claimed method to operate correctly across intermediate representation (IR) Block boundaries, each IR block must have an unambiguous representation of the current state (i.e., subsequent conditions) of the subject processor register as represented by the abstract registers.

By way of example, to illustrate the importance of determining the subsequent conditions and not simply 'entry points,' Applicant refers to Figure 6 and its respective description on pages 26-27 of the present specification. In Figure 6, IR Block 2 has two possible successors, either IR Block 3 or back at the beginning of IR Block 2. The route between IR Blocks 2 and 3 is shown with an arrow labeled as 'a'. The route from the end back to the beginning of IR Block 2 is shown as a dotted line labeled 'b' (a dotted line is used since, although this route exists it has not yet been traversed in the current execution of the translated program). On the initial translation of the corresponding program code when initially entering IR Block 2, IR Block 2 would be generated with a 'prevailing set of conditions.' If during the execution of the translated program, IR Block 2 were to branch back to itself along route 'b', the abstract register states it propagates would be incompatible with the abstract registers states which were originally passed to IR Block 2 by IR Block 1 because the states were altered during the execution of IR Block 2. If the *subsequent conditions* of the abstract register states were not analyzed and if branch back along route 'b' to the 'entry point' of IR Block 2 was simply performed, the subsequent re-execution of IR Block 2 would be incorrect. For the correct operation of the invention across IR Block boundaries, each IR Block must have an *unambiguous* representation of the current state of the subject processor register (as represented by the abstract registers). To overcome this problem, a new IR Block would be generated for the same portion of subject processor code according to the subsequent conditions . Thus, a Basic Block of subject processor code is represented using multiple intermediate representations, one intermediate representation for each different subsequent condition.

A. The Combination of Walters and Davidson Fails to Teach or Suggest Generating Additional Intermediate Representations for the Same Program Code

The combination of Davidson and Walters fails to teach or suggest generating different intermediate representations for the same "given portion" of program code in response to respective "previous prevailing" and "subsequent" conditions. Accordingly, the Office Action fails to establish a *prima facie* case of obviousness in setting forth the present rejection.

According to the Manual of Patent Examining Procedure § 2142:

> To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally the prior art reference (or references when combined) must teach or suggest all the claim limitations.

Applicant respectfully submits that the requirements for establishing an obviousness rejection have not been met. First, the prior art reference must teach or suggest all the limitations of the claims. See *In re Wilson*, 165 U.S.P.Q. 494, 496 (C.C.P.A. 1970). Here, as recognized on page 10 of the Office Action, Davidson fails to disclose the claimed step of generating additional an intermediate representation required to execute the same portion of program code with the subsequent conditions when the same portion of the program code is subsequently entered. Further, Walters fails to teach or suggest the generation of an intermediate representation of program code in any manner. Since neither reference so much as suggests generating different intermediate representations for the same "given portion" of program code, it necessarily follows that the combination of Walters and Davidson must fail to teach or suggest Applicant's intermediate representation generation method. Accordingly, combining those references fails to create even a *prima facie* case of obviousness. Accordingly, Applicants respectfully submit that the rejection of independent claims 1 and 15 and their respective dependent claims should be withdrawn. Further, this feature of generating different intermediate representations or target code for the same portion of program code is further set forth in new dependent claims 27 and 28 submitted for examination on their merits.

B. Walters Fails to Suggest Generating Additional Intermediate Representations

As set forth above, it is recognized in the Office Action that Davidson fails to teach or suggest generating additional intermediate representations for the same portion of program code, where the Office Action relies upon the teachings of Walters in asserting that it suggests the modification of Davidson's teachings to generate additional intermediate representations of the same program code. However, as set forth in detail above, Walters only discloses making a single translation of a portion of program code after making a determination of whether target code for that same portion of program code had previously been generated, where there is no teaching or suggestion in Walters for generating additional target code for a given portion of program code in addition to the already initially translated target code when subsequently encountering the same portion of program code. Thus, there is no teaching or suggestion in any of the cited prior art references to modify the teachings of Davidson to generate additional intermediate representations for the same portion of program code. Accordingly, it is respectfully submitted that independent claims 1 and 15 and their respective dependent claims are separately patentable over the cited prior art of record and allowance of these claims is earnestly solicited.

C. Neither Walters Nor Davidson Teaches or Suggests Examining the Subsequent Conditions When a Portion of Program Code is Subsequently Entered

Independent Claims 1 and 15 recite, inter alia, generating different intermediate representations for the same "given portion" of program code in response to respective "previous prevailing" and "subsequent" conditions. As set forth above, these conditions can be dynamic conditions that change during execution of the program code. Even assuming arguendo that the address of an entry point to a block of code could be considering a 'prevailing set of conditions,' the combination of Walters and Davidson at best still only discloses generating a single intermediate representation on an initial translation of a given portion of program code. The Examiner contends that the address of the entry point constitutes both a prevailing condition and a subsequent condition (see page 3 of Office Action). However, this reasoning fails because there is no situation in which the address of an entry point to a block of code when subsequently encountered would ever yield an additional intermediate representation in the combination of Walters and Davidson. In other words, when that same portion of program code is subsequently

entered, the combination of Walters and Davidson merely teaches to use the same intermediate representation that has already been generated, where there is no teaching or suggestion to determine the subsequent conditions and generate an additional intermediate representation for the same portion of program code based on these subsequent conditions. For these additional reasons, the rejection of independent claims 1 and 15 and their respective dependent claims is requested to be withdrawn.

DEPENDENT CLAIM 2

With respect to dependent claim 2, the Office Action relies on the assertion that Davidson discloses the feature of "generating an Intermediate Representation Block (IR Block) of intermediate representation for each Basic Block of program code as it is required by the program, each IR Block representing a respective Basic Block of program code for a particular entry condition," citing Col. 7, line 66 to Col. 8, line 8 of Davidson.

Claim 2 is dependent upon claim 1, thus each of the claim limitations of claim 1 must be read into claim 2. Therefore, claim 2 recites that an IR Block is generated on an initial translation of a given portion of program code for a particular entry condition and that an additional IR Block is generated for a subsequent particular entry condition when the same portion of program code is entered. As described on page 27 of the present specification, the claimed method represents a Basic Block of subject processor code using more than one IR Block with different entry conditions. The IR Blocks which are used to represent a single Basic Block with different entry conditions are referred to as IsoBlocks in the specification, where each IsoBlock is a representation of the same Basic Block of subject processor code, but under different entry conditions. The resulting IR Blocks and IsoBlocks of intermediate representation are advantageous in that they are less complex than an intermediate representation which is capable of representing all entry conditions, and may therefore be optimized more quickly and will also be translated into target processor code which executes more quickly.

Again, there is no disclosure in Davidson of generating different IR Blocks for the same Basic Block of program code based on different entry conditions. Therefore, Applicant respectfully submits that claim 2 is separately patentable over the cited combination of prior art, and the rejection of claim 2 should be withdrawn

DEPENDENT CLAIM 3

With respect to dependent claim 3, the Office Action relies on the assertion that Davidson discloses the feature of generating and storing special-case intermediate representation of a particular subject code instruction only for the functionality required at that iteration and at each subsequent iteration of the same subject code instruction, citing Col. 7, lines 24-65 and Col. 8, lines 9-17 of Davidson. Applicants further respectfully submit that the rejection of dependent claim 3 as obvious in view of Davidson et al and Walters et al. is also not well-founded.

Initially, Applicant notes that the teachings of Walters are again mischaracterized in its application to claim 3, where Walters' cited disclosure of "determining whether code has been generated and stored in a table, and if it has not, subsequently generating that code" only generates code for that instruction upon initially encountering that instruction. At each subsequent iteration of the same subject code instruction, Walters teaches that native code will have already been generated and stored in a table and Walters fails to teach or suggest generating any additional native code, whether special-case or not, for that same instruction after the target code is generated initially. Thus, the asserted combination of Davidson and Walters fails to establish a *prima facie* case of obviousness of claim 3.

Further, the features of claim 3 overcome a problem associated with emulation systems, namely the translation of unnecessary features of subject processor code. When a complex instruction is decoded from a subject processor code into the intermediate representation, it is common that only a subset of the possible effects of that instruction will ever be used at a given place in the subject processor program. For example, in a CISC (Complex Instruction Set Computer) instruction set, a memory load instruction may be defined to operate differently depending on what type of descriptor is contained in a base register (the descriptor describes how information is stored in the memory). However, in most programs only one descriptor type will be used by each individual load instruction of that program. A translator in accordance with claim 3 will generate special-case intermediate representation which includes, for example, a load instruction defined for only that descriptor type.

Davidson is relied upon in the Office Action as disclosing this feature of claim 3 of generating and storing special-case intermediate representation of a particular subject code

instruction. However, Davidson is similar to the prior art emulation systems described above in that its tuples appear to contain all of the features of the corresponding subject processor code, whether necessary or unnecessary. There is no teaching or suggestion in Davidson of generating and storing special-case intermediate representation of a particular subject code instruction only for the functionality required at that iteration. Accordingly, Applicant respectfully submits that dependent claim 3 is separately patentable over the cited prior art of record.

CLAIMS 4-6

With respect to dependent claim 4-6, these claims are further directed to methods in which a test procedure is performed to determine on subsequent iterations of a respective subject code instruction whether the required functionality is the same as that already represented by the associated stored special-case intermediate representation.

Again, Applicants respectfully submit that the rejections of dependent claims 4-6 as obvious in view of Davidson et al and Walters et al. are not well-founded. The Office Action cites to column 13, lines 7-44 of Davidson as disclosing the feature of determining on subsequent iterations of a respective subject code instruction whether the required functionality is the same as that already represented by the associated stored special-case intermediate representation. However, this cited portion of Davidson merely teaches determining whether two distinct expression tuples are guaranteed to compute the same value. Davidson fails to teach or suggest expression tuples having a plurality of functions or effects, where only a portion of those functions are generated and stored in a special-case intermediate representation of a particular subject code instruction only for the functionality required at that iteration of the instruction. As set forth above, Davidson is similar to the prior art emulation systems in that its tuples appear to contain all of the features of the corresponding subject processor code, whether necessary or unnecessary. Davidson's disclosure of determining whether two distinct expression tuples compute the same value is quite different from the associated test procedure recited in the present claims in which a determination is made on each iteration of a particular subject code instruction whether a special-case intermediate representation has been generated for the required functionality of that particular subject code instruction at that iteration, where the functionality is appropriately selected from a plurality of possible functionalities of the subject

code instruction.

Applicant respectfully submits that Davidson plainly fails to teach or suggest this associated test procedure recited in the claims, and Applicants respectfully submit that the rejection of dependent claim 4-6 should be withdrawn.

CLAIMS 11-12 AND 16

Applicant respectfully traverses the rejections of independent claims 11 and 16 as being obvious over Davidson in view of Walters and submits that the claims at issue are patentable over those patents for the following reasons. Independent claims 11 and 16 recite that when a block of program code can have alternative unused entry conditions or effects or functions, the intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions.

A. The Combination of Walters and Davidson Fails to Teach or Suggest Blocks of Program Code Having Unused Conditions or Functionality

The Office Action cites to column 7, line 66 to column 8, line 8 of Davidson as disclosing that the feature that at least one block of program code can have alternative unused entry conditions or effects or functions where the intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions. However, the cited portion of Davidson provides no mentioning of block of program code can have alternative unused entry conditions or effects or functions. Still further, Davidson fails to teach or suggest generating an intermediate representation having only those conditions, effects or functions out of a plurality of different possibilities that are required to execute that block of program code with a then prevailing set of conditions. Davidson appears to generate the same tuples for a given block of code no matter what conditions are present.

Applicant respectfully submits that the cited portion of Davidson plainly fails to constitute a disclosure of generating only that intermediate representation which is necessary to execute a block of program code with a then prevailing set of conditions. Accordingly, Applicant respectfully submits that the combination of Davidson and Walters fails to teach or suggest each and every claim element of independent claims 11 and 16 and cannot be used to

maintain a *prima facie* case of obviousness against these claims. Applicants submit that the rejection of independent Claim 11 and Claim 12 dependent therefrom should be withdrawn, and that those Claims should be allowed. Since Claim 16 was rejected on the same basis as Claim 11, the rejection of Claim 16 should also be withdrawn.

New claims 29 and 30 further set forth the details of this aspect of Applicant's invention by specifying that only a portion of the plurality of possible alternative entry conditions or effects or functions of the block of program code are represented in the intermediate representation as required to execute that block of program code with the prevailing set of conditions, where the intermediate representation does not contain any un-used entry conditions or effects or functions.

INDEPENDENT CLAIM 14

Applicant's independent Claim 14 recites a method of dynamically translating program code wherein the recited steps (a) - (c) are repeated in real time. Steps (a) – (c) include the steps of generating an intermediate representation of a block of first program code and then generating a block of second program code from the intermediate representation.

A. There Is No Motivation to Combine the Teachings of Walters and Davidson

The Office Action proposes to meet the terms of Claim 14 by combining the teachings of Walters and Davidson. In particular, the Office Action asserts that one skilled in the art would have been motivated to generate an intermediate representation as taught by Davidson during the run-time translation process of Walters. The asserted basis for such motivation is "for the purpose of optimizing the program code regardless of the programming language." Applicants respectfully submit that the evidence of record demonstrates that there would be no motivation to combine the references and that the motivation proposed by the Office Action is not supported by the record.

First, it will be appreciated that Davidson discloses what is known as a "static compiler." Such a compiler completely translates a particular program into another language before execution. The amount of time taken to perform such a translation is therefore relatively unimportant when compared to methods of dynamic translation such as that addressed by

Applicant's Claim 14.

On the other hand, Walters does relate to a dynamic run-time translator. Walters generally discuss prior art interpreters which ran 10-20 times slower than native speed, and the Walters disclosure itself is generally directed to methods for speeding up run-time execution. Therefore, one skilled in the art would clearly not be motivated to add an intermediate representation generation ("IR") layer into the Walters translator because of the prospect that the added complexity and overhead would tremendously slow down the Walters cross-compilation system. The possibility of "optimization" would further not constitute a motivating factor because Walters discloses that optimization is achieved within the context of his own non-IR based system (See e.g. Col. 3, lines 1-9).

Applicant notes that the prior art relied upon, coupled with the knowledge generally available at the time of the invention, must contain some suggestion or incentive that would have motivated the skilled artisan to modify a reference or to combine references. *See ATD Corp. v. Lydall, Inc.*, 159 F.3d 534, 48 U.S.P.Q.2d 1321 (Fed. Cir. 1998): *In re Oetiker*, 977 F.2d 1443, 24 U.S.P.Q.2d 1443 (Fed. Cir. 1992); *Symbol Tech., Inc. v. Opticon, Inc.*, 935 F.2d 1569, 19 U.S.P.Q.2d 1241 (Fed. Cir. 1991); *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1598 (Fed. Cir. 1988).

The mere fact that something can be modified or combined is not enough for an obviousness rejection. *See In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1990); *In re Gordon*, 733 F.2d 900, 221 U.S.P.Q. 1125 (Fed. Cir. 1984). The Federal Circuit has stated that the "obvious to try" standard is not the correct standard to apply. *See In re Roemer*, 258 F.3d 1303, 59 U.S.P.Q.2d 1527 (Fed. Cir. 2001); *In re Dow Chem. Co.*, 837 F.2d 469, 5 U.S.P.Q.2d 1529 (Fed. Cir. 1988).

It is also impermissible to use hindsight to determine obviousness. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1600 (Fed. Cir. 1988)(stating "One cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention.").

Here, there is no teaching or suggestion in either Walters or Davidson to generate a first block of program code into an intermediate representation dynamically when translating the first block of program code into a second block of program code. As set

forth above, Davidson discloses a "static compiler" while Walters discloses a dynamic translator directed to methods for speeding up run-time execution. Thus, one skilled in the art would not be motivated to utilize the complexity and overhead associated with Davidson's static compiler to tremendously slow down the Walters cross-compilation system.

This combination of prior art is impermissibly made through hindsight of Applicant's invention in an attempt pick and choose among isolated disclosures in the Walters or Davidson prior art references to recreate the claimed invention. The Examiner's hindsight is clearly illustrated in the paragraph extending between pages 4 and 5 of the Office Action in which the Examiner identifies Applicant's combination of generating an intermediate representation with its method of dynamically translating program code as support for the Examiner's motivation for combining Davidson's techniques for generating an intermediate representation with Walters' dynamic translator. Applicant's success in designing a method that allows dynamic translation to be achieved through the use of intermediate representations cannot be used as the impermissible hindsight motivation for combining the teachings of Walters and Davidson. It is only through the many features of Applicant's invention that such a combination of features provides desirable advantages, such as only generating intermediate representations required to execute a portion of program code with the then existing prevailing or subsequent conditions.

Accordingly, none of the criteria for an obviousness rejection have been met by the Office Action in setting forth the present rejection. Notably, the prior art references fail to teach or suggest all the claimed limitations, alone or even when combined. Furthermore, the Office Action fails to establish the existence of a suggestion or motivation to modify the reference teachings to arrive at the presently claimed invention without the hindsight of Applicant's teachings. Thus, Applicant respectfully submits that a *prima facie* case of obviousness has not been made out, and respectfully solicits allowance of Claim 14.

Conclusion

In view of the foregoing remarks and amendments, Applicants respectfully submit that the subject application is in condition for allowance. Applicants, therefore, respectfully request reconsideration and early notice of allowance.

Respectfully submitted,

PAUL, HASTINGS, JANOFSKY & WALKER LLP

Dated: May 13, 2005

By: _____
Bradley D. Blanche
Reg. No. 38,387

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. Box 919092
San Diego, CA 92191-9092
Phone: (858) 720-2500
Fax: (858) 720-2555